Daily recap

Final goal of the project

Comandare I droni con il 5G al posto del WIFI. Nessuno lo aveva mai fatto (work in progress) perche' bisogna dimostrare che il 5g soddisfi I requisiti minimi di sicurezza del drone. Degli esempi pratici, nel nostro specifico caso, sono i seguenti:

- multipli droni in volo: inviare un messaggio a tutti in modo tale che effettuino il landing, c'e il problema che bisogna schedulare I messaggi in maniera individuale (facendolo sequenzialmente, in parallelo..) e utilizzare le risorse in modo ottimale perche' ci sono altri utenti esterni (sensori, robot, cellualri) che stanno utilizzando la rete. Questo traffico puo interferire con I nostri comandi di emergenza e serve **un'intelligenza che dia la priorita' ad un traffico (tipo l'emergency landing) piuttosto che un altro. Piu in generale, serve che un intelligenza gestisca le risorse in modo tale che la rete riesca ad inviare questi pacchetti il piu veloce possibile.**

- I droni fanno cose ed inviano dati alla BS che inoltra dati ad un altro server (dove si trova Qground control. E.g un'azienda si occcupa della gestione dell'impianto e quindi Qground control non puo essere nella BS, ma e' al di fuori, nel cloud/server esterno) che potrebbe fare image processing; **tutto il processo di invio del segnale, elaborazione e ritorno deve avvenire in meno di 100ms** per mantenere I Qos . Inoltre il drone non deve mai perdere il segnale (anche perche siamo non line of sight). Che qualita' video/fps si utilizza per mantenere una latency bassa ma abbastanza buona?

- Open Ran, c'e' il drone ed una AI che ottimizza il collegamento (come detto prima) con la BS, di solito questa AI si fa runnare nella BS. Il problema e' che se il drone si sta spostando, quella AI la voglio spostare sulla nuova BS a cui il drone si e' collegato, quindi deve seguire il drone. Un'idea per fare cio e' di prendere un container con dentro la AI e appena il drone si sposta, ho un framework di orchstrazione che lo disinstalla dalla BS e lo reinstalla in un'altra (non possiamo installare l'AI in tutte le BS perche' sarebbe uno spreco di memoria).

- il drone deve capire da solo che sta uscendo dal coverage delle BS analizzando la qualita del segnale, e.g se sente che un segnale di una BS si attenua e invece di un'altra aumenta allora significa che e' ancora dentro la coverage area. Se invece sente solo un segnale che si attenua significa che sta uscendo dalla coverage area e il drone dovrebbe capire da solo che deve tornare indietro. L'analisi del segnale deve avvenire dentro il drone.

8 July++

Wireshark-mavlink packages:

frame len (#bytes) name of packet

110 COMMAND_LONG: MAV_CMD_COMPONENT_ARM_DISARM

88 COMMAND_ACK: MAV_CMD_COMPONENT_ARM_DISARM

110 COMMAND_LONG: MAV_CMD_NAV_TAKEOFF

88 COMMAND_ACK: MAV_CMD_NAV_TAKEOFF

84 SET_MODE: landing/maintaining position

88 COMMAND_ACK: MAV_CMD_DO_SET_MODE

MIssion upload:

82 MISSION_COUNT

115 MISSION_ITEM_INT: one for each of the waypoints (including takeoff and landing point)

[…]

109 MISSION_COUNT PING

82 MISSION_COUNT

Capture name	Packets Per Second (PPS)	Packet Size (PS)	Base Station (BS)		User Equipment (UE)	
			Packets Per Second (PPS)	Packet Size (PS)	Packets Per Second (PPS)	Packet Size (PS)
Noflight-1	460.9	114	97	66	363	127
Noflight-2	450.9	115	94.2	67	355.7	128
Noflight-3	452.8	115	89.3	67	362.6	126
Flight-1	456.3	115	95.5	67	359.6	127
Flight-2	437.8	117	84.3	67	352.5	129
Flight-3	430.1	118	86.6	67	342.2	130
AVERAGE	**448.1 = 448**	**115.7 = 116**	**91.2 = 91**	**66.8 = 67**	**355.9 = 356**	**127.8=128**

MEETING WITH SALVO-*77-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*7-*-/*-/*-/*-/*-/*-/*-/*-/*-/*-/*-/*-/*--+

Threshold 100ms = requirement dato dal 3GPP\ETSI (che e’ il 3gpp europeo) per avere un “beyond flight on sight”.

Come creare il modello di traffico: prendo tutti I pacchetti di telemetria (frequenza al secondo e la loro grandezza) e li riprodiciamo su Iperf.Idem per I nostri comandi.

Scenario Colosseum: 12352, 12356 – ci sono 3 UAV che si muovono a forma di 8 (Salvo ci puo dare il modello 3D) , 9 BS sul tetto (su una grid 3x3 e si attivano una alla volta) e 3 utenti mobili fissi.

Frequenze: utilizzare le frequenze vicino al wifi altrimenti utilizzare le frequenze a banda 7 a bassa potenza (80 gain va bene)

28 June

Next steps: study pckt size and latency for the following commands: t**ake off, stop and fly, change waypoint on going, land**

We have to save latency and pckt size for them. We have to do different measurements with different distances: on the first bench, second bench, third bench.

How to get pckt size and latency? Wireshark, QGC (check if the sent pckt is saved on log file with a timestamp)

Save also log files from SCOPE.

THEN: once we created the traffic model, reproduce it on arena or colosseum to obtain more data point.

27 june

Actual flow to follow:

GNB:
bash BaseStation.sh
tmux
bash GnBrun.sh

UE:
Bash UE.sh
Tmux
bash uerun.sh

24 june

We want to send packets from the GnB host to the UE host: to do so we have to setup **ip tables** to being able to send pckts until the UE host (to the Bridge 240.84.80.1) and allows to answer to the GnB host (to the bridge 10.241.115.1).
Moreover, the 4g LTE doesn't allow a communication from a GnB and a host that is after the UE, the only possible final destination is the UE (the effect is called GTP tunnel). GTP is used to encapsulate user data when passing through core network: once the pckt is encapsulate, the GNB check the Ip of the incapsulated pckt and check if the ip match with any of the connected UE. If not the packet is discarder

(unknown ip) otherwise it is sent to the correct UE.
To solve this problem (because we are sending pckts to a network interface which is not the UE), we have to change the destination address of the pckt inside the GnB and inside the UE (before the encapsulation) using **DNAT** (http://linux-ip.net/html/nat-dnat.html). The generated pckt is originally sent to the bridge 240.84.80.1, the DNAT inside the GnB changes the into 172.16.0.8, then once the pckt arrives to the UE the DNAT change it again to the real ip, 240.84.80.1. In conclusion we have to add 2 DNAT rules: one in the GnB and one in the UE.
Finally, in the UE host, Mavink router ha to be active in order to open the port for QGC and allowing the answer for command pckts.

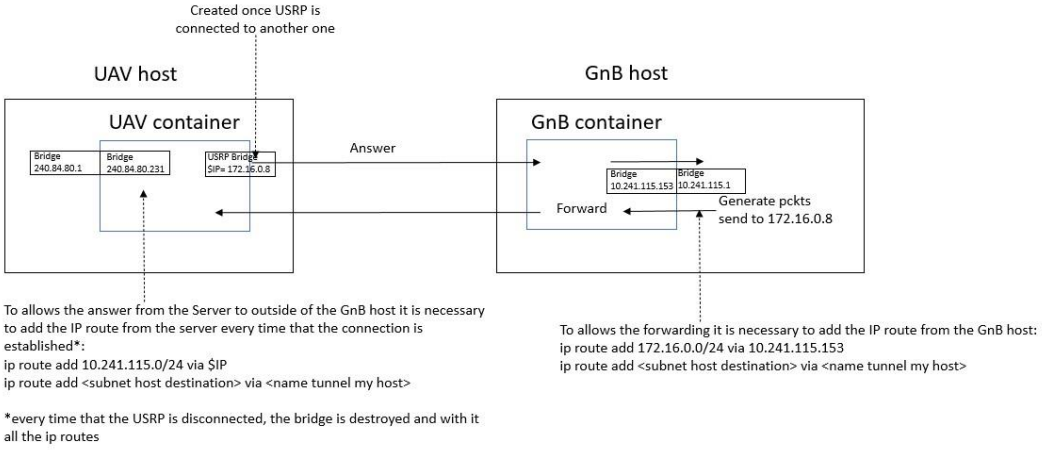
21 june

Next step: Say to QGroundControl to check for the drone ip on another network interface: the USRP one and not anymore the WIFI. So, we should be able to control the drone and check its information through the USRP (LTE) and not anymore WIFI.

17 June

Installed **Tmux** in both containers. For one of the containers, I had a problem so to lunch Tmux and I used the following script (nd saved as alias): **script -qfc tmux /dev/nul**
To exit from Tmux window: ctrl+a d
Open new tab: ctrl+a | or ctrl+a -
Kill every session: pkill -f tmux
Kill singe session: ctrl+a x

15 june
Allows forwarding of pckts with "IP route add..."



I performed **ip route add <subnet-container-UE> via <name-tunnel-hostPC>** on my PC (no container)
Then on the container UE I performed **ip route add <subnet-bridge-hostPC> via <name-interface-UE>** .
NB this command ha to be performed every time on the UE. A script has been wirtten

Useful commands:

Tcpdump -i <name-interface> = name-interface can be obtained thanks to **ifconfig**, tcpdump is a command useful to check information about received and sent pckts (similar to Wireshark but on terminal). Useful to use on server side.

iproute add <subnet-host-destination> via <name-tunnel-my-host>: necessary to do every time that the UE is closed because the interface is desroyed (only on UE side).

6 june

Commands recap for the base station and user equipment:

- BS: epcRun.sh, then enbRun.sh (in two differnet terminals)
- UE: ueConf.sh

26 may

https://github.com/wineslab/colosseum-scope/blob/9c3e0e82915d63f6d1d036cb43a06aef6a3325b2/radio_code/srslte_config/user_db.csv : this file contains every IP mapped with NUID

TO DO:

1) script to obtain the ip address: we know that the BS once it connects to the new user returns the **UE IMSI:** and so, we can modify the original file to return or also execute the iperf command.

2) permettere il forwarding tramite da porta A a porta B all'interno del container della BS e che il traffico sia duale (anche da B a ad A) -> iptables/netconf/nat

3) faccio ping

25 may
BS: sudo srsepc epc.conf

BS: sudo srsenb enb.conf

UE: sudo srsue ue.conf

iperf is working only if performed inside containers. There is no communication outside them
Possible solution: resinstall scope with Docker and allow bridge network

24 may

Talk to Salvo about next steps: configure the new radio? Or do the Wireshark experiment? What's first

23 may

Allowed lxd real time processing

Stop container, open the following file "**lxc config edit clear-sloth**" and add "**limits.kernel.rtprio: "99"** " and **" security.privileged: "true"** "

Radio couldn't see each other : we changed frequencies from **enb.conf** and **ue.conf** files. We increased the gain from 20dB to 80dB and matched the frequencies of the UE to the one showed in the screen from the gnb. The gnb can force the frequency and so we have to change the Ue file.
In both file change from **colosseum** to **auto**.
Also change **scope_cfg.txt** and change colosseum_testbad from 1 to 0

19 may

lxc list

lxc exec <name_image> bash

Srsue runned in the drone:

To enabled the USB, run the following commands outside the container

lxc config set <container-name> raw.lxc "lxc.cgroup.devices.allow = c 189:* rwm"

lxc config device add <container-name> b210usb usb mode="0777"

And then restarted the container.

- Once the Container is started, run "**sudo srsue radio_code/srslte_config/ue.conf**"
- Saved the image with **lxc publish profound-shepherd --alias scope**

The script is running but keeps to search for a free cell, that has to be configured with a BS and a radio

Inside the container try "**uhd_find_devices"** and check if it sees the usb. In case install all the things that are written.

May 17

PROJECT OVERVIEW

QgroundControl---------- 4G radio [SCOPE] ---------- Companion computer [SCOPE]--FCU

1) Installo **SCOPE** sulla radio come BS, sul companion computer installo il container di SCOPE come UE
2) Utilizzo **Iperf** per testare l'invio dei comandi IPERF lo install sulla NUC e sulla BS, uno dei due diventa il server e la BS generera' traffico che verra' Inviato dalla BS e ricevuto dall'UE.
   IPERF permette di generare pacchetti ed impostare una grandezza ed il bitrate: utilizzato per calcolare il throughput.
   QPERF utilizzato per analizzare la latency tra BS e UE.
   Come conoscere il traffic model? **Wireshark**: genero dei comandi tramite QgrounControl (utilizzando una logica, es takeoff, land, next path...) e li catturo. I pacchetti avranno sempre una certa grandezza (probabilmente variera' il sequence number) quindi controllero' quanto sono grandi e a che frequenza sono inviati in modo da emularla tramite IPERF/QPERF.
3) Stress test sulla batteria in %, mando una sequenza di comandi (es land e take off in sequenza) per capire quanta % viene utilizzata.
4) **Tunnel** QgroundControl: dei comandi di QGC entrano nell BS (con scope installato) e vengono inoltrati all'UE. In poche parole l'obiettivo e' fare quello che sta accadendo ora tramite WI-FI (swarm control) ma tramite 4G (in futuro il 5G). Suggerimento: cambiare l'indirizzo ip e porta di QGroundControl con quelli di SCOPE (del drone)
5) Filtro dei pacchetti dei comandi nel caso ci sia altro traffic nella BS: per farlo si utilizza l'indirizzo IP, ovvero si fa il check dell'IP nei pacchetti e se questi arrivano dal PC/Client con QGroundControl, vengono inoltrati all'UE (UAV/Companion computer sopra UAV).

LONG TERM GOAL: effettuare diversi tipi di test per:

- energy consumption
- latency in base a quanto ci spostiamo all'interno dell'ambiente indoor: i pacchetti di QGC da quando vengono generate a quando vengono ricevuti devono passare max 100ms (limite di latency su UAV 5G). Se si verifica capire quando e come risolverla. Un modo per risolverla e' tramite network slicing: dare priorita' (dare tutte le risorse a quell traffico) a dei messaggi inviati da QGC.
- Handover tra 2 BS a Burlinghton mentre si mandano dei segnali di emergenza.
- Computer vision: CV e' eseguito in un server e quando UAV si muove vuole cambiare delle BS (diversa dal CV server) per mantenere un segnale con gli UAV. Sarebbe da trovare un modo per mantenere il segnale con il CV server anche se la BS cambia. Il problema e' che ci sono numerosi droni che si connettono contemporanemaente e non va bene se si connettono tutti allo stesso CV server. E' un problema di ottimizzazione per l'handover della BS.
- Come inviare I video al CV server per l'elaborazione: 4k, full hd, qualche frame

Connect to Colosseum through SSH

ssh file-proxy in the home directory

Pwd1: Casper05

Pwd2: Casper05

May 12

Definitions

Mavlink: messaging protocol for communicating with drones (and between onboard drone components). **Mavlink router** is Installed inside the companion computer (Linux or Raspberry).

Qground control: is a program (Linux) that provides full flight control and mission planning for any MAVLink enabled drone. Usefull to send commands to the drone and see what it is happening inside it.

Px4 – Autopilot (software): PX4 autopilot is an open-source autopilot system installed inside the FCU (Flight control unit) of the drone that allows the drone to fly once every sensor is calibrated correctly.

Ardupilot: An alternative of PX4, ArduPilot is an open source, unmanned vehicle Autopilot Software Suite, capable of controlling autonomous UAVs.

Pixhawk 4 (hardware): model of the FCU.

Dronekit: API allows developers to create Python apps that communicate with vehicles over MAVLink.

Gazebo: software simulator for drones

Jmavsim: as Gazebo but a simpler multirotor/Quad simulator

Future steps:

local pc is a client sending commands to a BS (arena radio for 5G) (in this moment we are using swarmcontrol, the local router, using WIFI) that will forward each command to the drone. To do so we have to work with SCOPE (to study).
Talk to Salvo to check if this idea is correct and to define better the future steps of the project: what's next? Do I have to work alone or do I have to do a different work from Pietro?

May 4

To fly MONARCH drone, prameter changes from default:

ARMING_CHECK = GPS_LOCK

BRD_SAFETYENABLE = DISABLED

We managed to run **Hardware in the loop (HIL)** with **Jmav** and **QGroundControl:**

First of all we updated the software to **Px4** (not Ardupilot)

We followed this tutorial https://docs.px4.io/master/en/simulation/hitl.html to start HIL

May 2

To fly HADRIAN, prameter changes from default:

AHRS_GPS_MINSAT=0 (for inside flight)

AHRS_GPS_USE=DISABLE (for inside flight)

ARMING_CHECK = DISABLED

BRD_SAFETYENABLE = DISABLED

May1

Abhi drone: ssh pi@192.168.10.7

Pwd: wnesl

26 April 2022

- Installed Linux
- Cloned repositories and installed QGroundControl

To run simulator,

Go to **/Documents/PX4-Autopilot** and run the following command

bash ./Tools/jmavsim_run.sh -q -s -d /dev/ttyACM0 -b 921600 -r 250

(**-d /dev/ttyACM0 -b 921600 -r 250** is for USB/serial connection,

otherwise on WIFI **-i 192.168.10.X -p 5760**)

Then run **QGroundControl.AppImage**